THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

# COMP 110
# Introduction to Programming

**Tuesday September 30, 2014**

Jay Aikat
Fall 2014
TR 9:30 – 10:45, GS-G100

## Previous Class

- What did we discuss?

# Announcements

- **Assignment3 and Lab4 – deadlines extended**



# Primitive Types

| Type Name | Kind of Value | Memory Used | Range of Values |
|---|---|---|---|
| `byte` | Integer | 1 byte | –128 to 127 |
| `short` | Integer | 2 bytes | –32,768 to 32,767 |
| `int` | Integer | 4 bytes | –2,147,483,648 to 2,147,483,647 |
| `long` | Integer | 8 bytes | –9,223,372,036,8547,75,808 to 9,223,372,036,854,775,807 |
| `float` | Floating-point | 4 bytes | $\pm 3.40282347 \times 10^{+38}$ to $\pm 1.40239846 \times 10^{-45}$ |
| `double` | Floating-point | 8 bytes | $\pm 1.79769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$ |
| `char` | Single character (Unicode) | 2 bytes | All Unicode values from 0 to 65,535 |
| `boolean` | | 1 bit | True or false |

## Exercise 1

- Ask user to input an integer
- If input is a one-digit number, then print the product of three consecutive integers, starting with the input
- If input is a two-digit number, then print the square of the number
- If input is a three or more digits, then quit the program with a message to user
- Setup a loop for the above to run this 5 times



## Exercise 2

- Ask user to input an integer
- If input is a one-digit number, then print the product of three consecutive integers, starting with the input
- If input is a two-digit number, then print the square of the number
- If input is a three or more digits, then quit the program with a message to user
- Continue if the user says “yes” to the question: “Do you wish to continue”?

## Brackets

- Consider this:

2 * 5 + 7

2 * (5 + 7)

2 * (5 + (7 * 6))

2 * (5 + (7 * (6 – 3)))



## Local Variables

- Open Eclipse
- New Java project etc.. You know the drill!

```
public class test123 {
        public static void main(String[] args)
        {
                int  num1 = 5;
                int  count;

                for (count = 0; count <= num1; count++){
                        System.out.println(count);
                }
        }
}
```

# Local Variables

```
public class test123 {

        public static void main(String[] args)
        {

                int num1 = 5;
                int count;

                for (count = 0; count <= num1; count++) {
                        int num2 = 10;
                        System.out.println(count);
                         System.out.println(num2);
                }
        }
}
```



# Local Variables

```
public class test123 {

        public static void main(String[] args)
        {

                int num1 = 5;
                int count;

                for (count = 0; count <= num1; count++) {
                        int num2 = 10;
                        System.out.println(count);
                         System.out.println(num2);
                }
                System.out.println(num2);
        }
}
```

## The Class `String`

- We've used constants of type `String` already.
  `"Enter a whole number from 1 to 99."`
- A value of type `String` is a
  - Sequence of characters
  - Treated as a single item.



## String Methods

- An object of the `String` class stores data consisting of a sequence of characters.
- Objects have methods as well as data
- The `length()` method returns the number of characters in a particular `String` object.
- Try this:

```
String greeting = "Hello";
int n = greeting.length();
System.out.println("Length of the string is " + n);
```

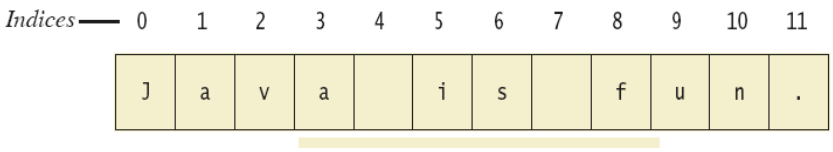
## The Method `length()`

- The method `length()` returns an `int`.
- You can use a call to method `length()` anywhere an `int` can be used.

```
int count = command.length();
System.out.println("Length is " +
    command.length());
count = command.length() + 3;
```



## `String` Indices




- Positions start with 0, not 1.
  - The 'J' in "Java is fun." is in position 0
- A position is referred to an an *index.*
  - The `'f'` in `"Java is fun."` is at index 8.

# `String` Methods

| |
|---|
| `charAt(`*Index*`)`<br>Returns the character at *Index* in this string. Index numbers begin at 0. |
| `compareTo(`*A_String*`)`<br>Compares this string with *A_String* to see which string comes first in the lexicographic ordering. (Lexicographic ordering is the same as alphabetical ordering when both strings are either all uppercase letters or all lowercase letters.) Returns a negative integer if this string is first, returns zero if the two strings are equal, and returns a positive integer if *A_String* is first. |
| `concat(`*A_String*`)`<br>Returns a new string having the same characters as this string concatenated with the characters in *A_String*. You can use the ⇓ operator instead of `concat`. |
| `equals(`*Other_String*`)`<br>Returns true if this string and *Other_String* are equal. Otherwise, returns false. |

**Methods and their return types…**



# `String` Methods

| |
|---|
| `equalsIgnoreCase(`*Other_String*`)`<br>Behaves like the method `equals`, but considers uppercase and lowercase versions of a letter to be the same. |
| `indexOf(`*A_String*`)`<br>Returns the index of the first occurrence of the substring *A_String* within this string. Returns -1 if *A_String* is not found. Index numbers begin at 0. |
| `lastIndexOf(`*A_String*`)`<br>Returns the index of the last occurrence of the substring *A_String* within this string. Returns -1 if *A_String* is not found. Index numbers begin at 0. |

## `String` Methods

| |
|---|
| `length()`<br>Returns the length of this string. |
| `toLowerCase()`<br>Returns a new string having the same characters as this string, but with any uppercase letters converted to lowercase. |
| `toUpperCase()`<br>Returns a new string having the same characters as this string, but with any lowercase letters converted to uppercase. |



## `String` Methods

| |
|---|
| `replace(`*OldChar*, *NewChar*`)`<br>Returns a new string having the same characters as this string, but with each occurrence of *OldChar* replaced by *NewChar*. |
| `substring(`*Start*`)`<br>Returns a new string having the same characters as the substring that begins at index *Start* of this string through to the end of the string. Index numbers begin at 0. |
| `substring(`*Start, End*`)`<br>Returns a new string having the same characters as the substring that begins at index *Start* of this string through, but not including, index *End* of the string. Index numbers begin at 0. |
| `trim()`<br>Returns a new string having the same characters as this string, but with leading and trailing whitespace removed. |

## Exercise 3

- Ask user to input a string
- Find the length of the string
- If the string length is less than 10, find the 5th character in the string, else find the 11th character
- Compare this string with another string
- Replace a character within a string
- Find a substring within the string



## Next class (Thu, Oct 2)

- More on String